TCP- und UDP-Sockets in Java

Grundlegende Mechanismen

• Server reserviert Port:	Klient: -	Server: **bind**
• Server nimmt Verbindungswünsche an	Klient: -	Server: **listen**
• Klient möchte sich verbinden	Klient: **connect**;	Server: **accept**
• Datenaustausch	Klient/Server: **read/write**	
• Ende der Kommunikation	Klient/Server: **close**	

Sockets in Java

- **java.net.Socket**
  - Ein Socket ist ein Endpunkt einer Verbindung.
  - Socket wird gebunden an eine Adresse und Port auf dem entferntem Rechner und an ein Port auf dem lokalen Rechner
- Konstruktor
  - **Socket socket = new Socket("www.vs.uni-kassel.de", 80);**
  - Nur für Klientenseite
- Methoden
  - **public InputStream getInputStream()**
  - **public OutputStream getOutputStream()**
  - **public void close()**

Beispiel

- Programm das ein paar Daten zu „www.spiegel.de“ schickt und die Antwort auf der Konsole ausgibt (ohne Exception-Verarbeitung)

```
Socket socket = new Socket("www.spiegel.de",80)
OutputStream out = socket.getOutputStream();
InputStream in = socket.getInputStream();
String sMessage = "GET / HTTP/1.1\nHost:www.spiegel.de\n\n";
byte[] aby = sMessage,getBytes();
for (int i=0;i<aby.length;i++) out.write(aby[i]);
out.flush();
int read = in.read();
while(read!=-1) {
    System.out.print(read);
    read=in.read();
}
in.close(); out.close(); socket.close();
```

Serverseite in Java

- java.net.ServerSocket
  - „Warten auf Verbindungen“
  - Konstruktor
    ServerSocket ssocket = new ServerSocket(1000);
  - Methoden
    public Socket accept()

Beispielserver

- Echoserver
  - Empfangenen Daten auf Konsole ausgeben und zurückschicken

```
ServerSocket ssocket = new ServerSocket(7);
Socket socket = ssocket.accept();
InputStream in = socket.getInputStream();
OutputStream out = socket.getOutputStream();
int read = in.read();
while(read!=-1) {
    out.write(read);
    System.out.print(read);
    read = in.read();
}
out.flush();
in.close();out.close();socket.close();ssocket.close();
```

UDP-Sockets

- Socket binden
  - **Klient/Server: bind**
- Senden und empfangen
  - **Klient/Server: send, receive**
- Sockets schießen
  - **Klient/Server: close**

DatagramSocket

- Konstruktoren
  - Gebunden an beliebigen Port: **DatagramSocket()**
  - sonst: **DatagramSocket(int port)**
- Verbindung
  - **void send(DatagramPacket p)**
    - Die Zieladresse steht im DatagramPacket
  - **void receive(DatagramPacket p)**
- Socket schliessen
  - **void close()**

DatagramPacket

- Konstruktoren
  - **DatagramPacket(byte[] buf, int length)**
    - Paket der Länge length konstruieren
  - **DatagramPacket(byte[] buf, int length, InetAddress address, int port)**
    - Paket konstruiren und „adressieren“
- set- und get-Methoden
  - **void setData(byte buf)**
  - **byte[] getData()**
  - **void setAddress(InetAddress iaddr)**
  - **InetAddress getAddress()**

UDP-Beispiel

- Server wartet auf ein Paket mit Zeichenkette und antwortet mit einer anderen Zeichenkette

UDP-Beispiel: Server

```
public UDPPongServer(int portNo) throws java.io.IOException {
  byte[] inData = new byte[1024]; // Platz für Pakete vorbereiten
  byte[] outData = new byte[1024];
  String message;
  DatagramSocket socket = new DatagramSocket(portNo); // Socket binden

  while (true) {
     // Ein Paket empfangen
     DatagramPacket in = new DatagramPacket(inData,inData.length);
     socket.receive(in);
     // Infos ermitteln und ausgeben
     InetAddress senderIP = in.getAddress();
     int senderPort = in.getPort();
     message=new String(in.getData(),0,in.getLength());
     System.out.println("Got "+message+" from "+senderIP+","+senderPort);
     // Antwort erzeugen
     outData = "Pong".getBytes();
     DatagramPacket out = new DatagramPacket(outData,outData.length, senderIP,senderPort);
     // Antwort senden
     socket.send(out);
  }
}
```

UDP-Beispiel: Klient

```
public UDPPingClient(int portNo) throws java.io.IOException {
    byte[] inData = new byte[1024];
    byte[] outData = new byte[1024];
    String message;
    // Socket erzeugen
    DatagramSocket socket = new DatagramSocket();
    // Paket bauen und adressieren
    InetAddress serverIP = InetAddress.getByName("localhost");
    outData = "Ping".getBytes();
    DatagramPacket out = new DatagramPacket(outData,outData.length, serverIP,portNo);
    // Paket senten
    socket.send(out);
    // Antwort empfangen und ausgeben.
    DatagramPacket in = new DatagramPacket(inData,inData.length);
    socket.receive(in);
    message=new String(in.getData(),0,in.getLength());
    System.out.println("Got "+message);
    // Socket schliessen
    socket.close();
}
```